Automatic IPv4 to IPv6 Transition D1.2 - Network representation and pre-requisites

Frédéric Beck, Isabelle Chrisment, Olivier Festor

Automatic IPv4 to IPv6 Transition D1.2 - Network representation and pre-requisites

Frederic Beck, Isabelle Chrisment, Olivier Festor

June 18, 2009

Contents

1 Introduction **2**

2 Network Representation **3**
2.1 Overview: the simplest case . . . 3
2.2 Network with a backbone . . . 4
2.3 Network with Loop . . . 6
2.4 Multihomed Network . . . 8
2.5 Multihoming at Site Level . . . 11
2.5.1 2 ISPs and 1 border router . . . 11
2.5.2 2 ISPs and 2 border routers . . . 11
2.5.3 Different Border Router . . . 13
2.6 Network with DMZ . . . 13
2.6.1 VLAN . . . 16

3 Pre-requisites to an IPv4 to IPv6 Transition **18**
3.1 Connection to the IPv6 World . . . 18
3.2 Network Topology . . . 18
3.3 Additional Network Related Information . . . 18
3.3.1 Routers . . . 19
3.3.2 VLANs . . . 19
3.4 Constraints . . . 19

4 Possible Sources of Information **21**
4.1 Automatic Network Discovery . . . 21
4.1.1 Hynetd . . . 22
4.1.2 Cheops-ng . . . 22
4.1.3 Netdisco . . . 22
4.1.4 Zenoss . . . 23
4.1.5 Homemade Cooking . . . 23
4.1.6 Conclusion . . . 23
4.2 CiscoWorks . . . 24
4.3 IPv4 Security Policy . . . 24

5 Conclusion **25**

Abstract

Over the last decade, IPv6 has established itself as the most mature network protocol for the future Internet. While its acceptance and deployment remained so far often limited to academic networks, its recent deployment in both core networks of operators (often for management purposes) and its availability to end customers of large ISPs demonstrates its deployment from the inside of the network leading to the edges.

For many enterprises, the transition remains an issue today. This remains a tedious and error prone task for network administrators.

In the context of the Cisco CCRI project, we aim at providing the necessary algorithms and tools to enable this transition to become automatic. In this report, we present the first outcome of this work, namely an analysis of the transition procedure and a model of target networks on which our automatic approach will be experimented. We also present a first version of a set of transition algorithms that will be refined through the study.

Chapter 1

Introduction

IP networks are widely spread and used in many different applications and domains. Their growth continues at an amazing rate sustained by its high penetration in both the Home networks and the mobile markets. Although often postponed thanks to hacks like NAT, the exhaustion of available addresses, and other scale issues like routing tables explosion will occur in a near future.

The IPv6 [1] was defined with a bigger address space (128 bits) and comes along with new built-in services (address autoconfiguration [6], native IPSec, routes aggregation, simplified structure...). It is a fact that IPv6 deployment is slower than foreseen. Many reasons are valid to explain this: economical, political, technological, and human. Despite this slow start, IPv6 is today more than ever the most mature network protocol for the future Internet. To faster its acceptance and deployment however, it has to offer autonomic capacity that emerge in several recent protocols in terms of self-x functions reducing end often eliminating the man in the loop. We are convinced that such features are also required for the evolutionary aspects of an IP network, the transition from IPv4 to IPv6 being an essential one.

In this project, we are interested in the scientific part of the technological problems that highly impact human acceptance. Many network administrators are indeed reluctant to deploys IPv6 because, first, they do not know well the protocol itself, and they do not have sufficiently rich algorithmic support to seamlessly manage the transition from their IPv4 networks to IPv6. To address this issue, we investigate, design and aim at implementing a transition framework with the objective of making it self-managed.

As the IPv4 to IPv6 transition is a very complex operation, and can literally lead to the death of the network, there is a real need for a transition engine to ease and secure the network administrator's task; the ideal being a "one click" transition.

This report presents the logical representation of the network on which our study will be based. We will first define this representation, and then give the representation of each topology considered in deliverable D1.1. Then, we will identify the possible sources of information that can be used in order to get all the information required to generate the network representation. Finally, we will give the pre-requisites to an IPv4 to IPv6 transition.

Chapter 2

Network Representation

In this study, we will use a logical representation of the network.

2.1 Overview: the simplest case

We chose to represent the network as a hierarchical, oriented graph.

This graph has one and only one root, which we will call *border router*, and which is the router connected to the IPv6 world. The leaves of the graph are the end-users subnets, which we call *end-networks*. The vertices between the root and the leaves can be routers, or inter-connection networks. An inter-connection is represented by a simple link if it connects only 2 routers, but if it connects more than 2 routers, it is represented by a specific network vertex we call *backbone*.

The simplest case that can be encountered is the tree topology of D1.1 shown in figure 2.1.

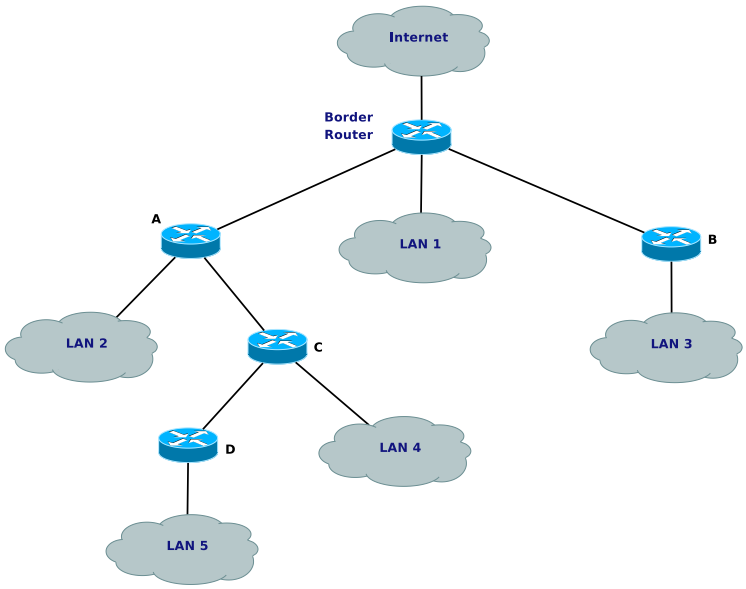

Figure 2.1: Simple Network as a Tree

This physical topology can be simply reused as logical representation of the network as it follows all the constraints detailed previously, as shown in figure 2.2.

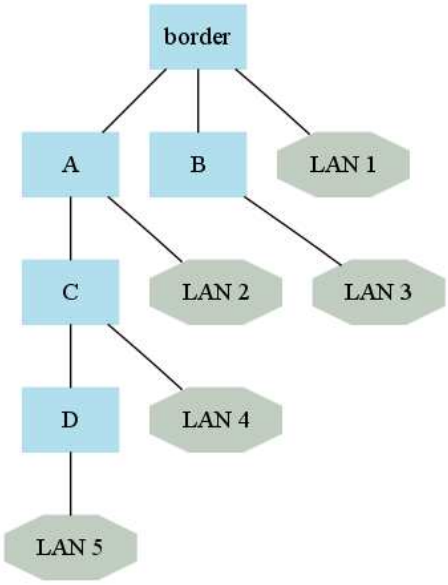

Figure 2.2: Logical Representation of a Tree Network

As we decided to not take into account the connection with the IPv6 world, assuming that this has been already configured and deployed, this interconnection is not represented, in order to have the border router as root of the graph.

Using this logical representation, all vertices need to find their shortest path to the root in order to determine the hierarchical relations in the graph. The gateway of a vertex will be by default the next vertex, if following the shortest path to the root. In the physical topology, it means that the gateway for the network *LAN 5* is the router *D*, and that the gateway of that router is the router *C*. In order to determine the shortest path, all links in the graph have been assigned the same weight, namely 1.

In the next sections, we will introduce complexity in the network topology. We will use the topologies defined in deliverable D1.1, and explain case by case, how the specificity of each topology is taken into account in the logical representation.

2.2 Network with a backbone

As we said, the intermediate vertices can be routers or inter-connection networks. An inter-connection network network can connect two or more routers. In that second case, it will be represented as a network. We take as example the physical network shown in figure 2.3, which consists in a tree topology with a backbone connecting 3 routers.

The logical representation of the network is shown in figure 2.4.

When running the shortest path algorithm on that logical representation, with all links weights set to 1, we obtain a distance of 2 between the routers *A* and *A* and the border router, while it is supposed to be 1. It means that for these two routers, their gateway is the backbone, and not the border router, which is not the expected outcome.

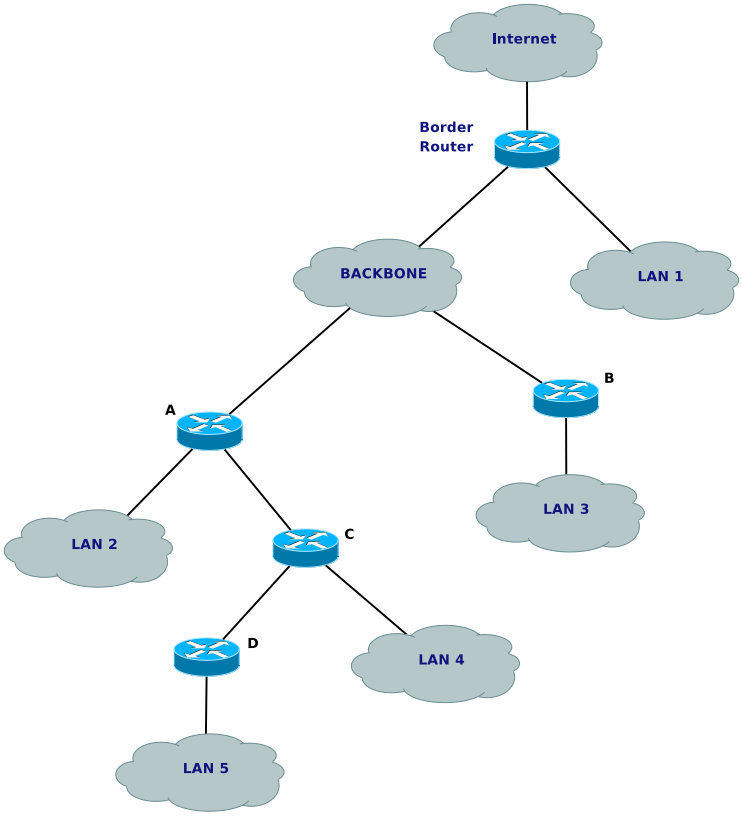

Figure 2.3: Network as a Tree with a Backbone

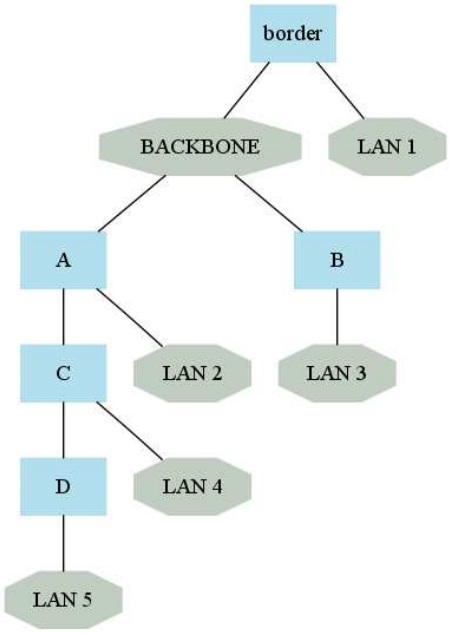

Figure 2.4: Logical Representation of a Tree Network with a Backbone

In this case, the weights need to be adapted to the physical topology. The link going from the border router to the backbone will thus keep its weight to 1, whereas the links going from the backbone to the routers A and B will have a null weight, as shown in figure 2.5.

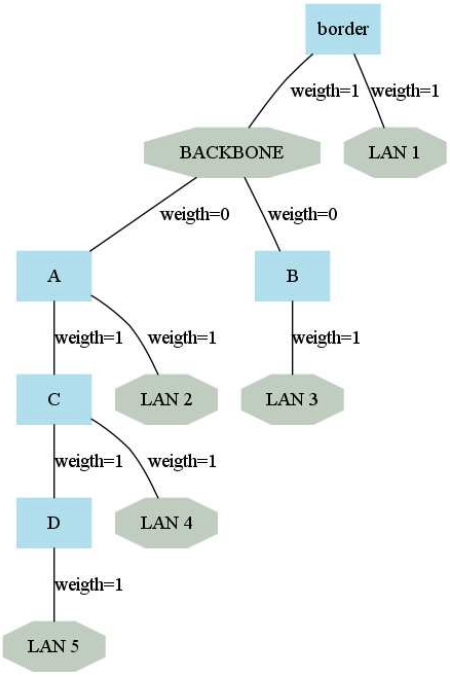

Figure 2.5: Logical Representation of a Tree Network with a Backbone and Corrected Weights

This time, if the shortest path algorithm is run, the precedence is respected and the correct paths are determined.

The weigth assignment follows 4 rules:

1. All links have a weigth which can be 0 or 1
2. All links between 2 routers have a weigth of 1
3. All links from a router to a network (leave or backbone) have a weigth of 1
4. All links from a network to a router have a weigth of 0

In the following section, we will not mention the weights, but assume that they follow these rules.

2.3 Network with Loop

In the physical topology, loops are accepted in the routing, with one subnet being connected to two routers at the same time, for redundancy issues usually. Figure 2.6 shows such a topology.

In this topology, the network *LAN 2* is connected to two different routers. The difficulty here is to define which router will be the primary upstream for the subnet, and which one will be the backup.

The network representation generated from this topology thus also has a loop, as shown in figure 2.7.

By following the weights rules defined earlier, both links from the routers to the network have a weigth set to 1. The shortest path algorithm gives two different paths for *LAN 2* to the border. The first one with a path length of 2 via the router A, and the second one via the router D with a length of 4. By

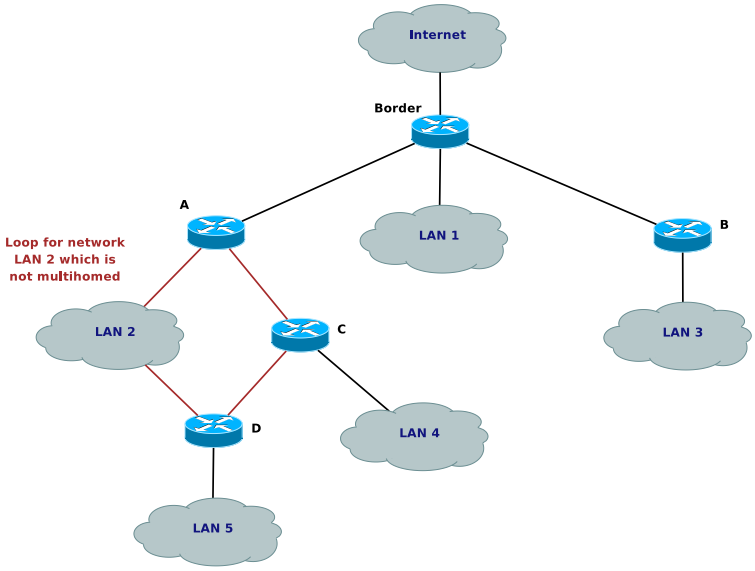

Figure 2.6: Network with a Loop

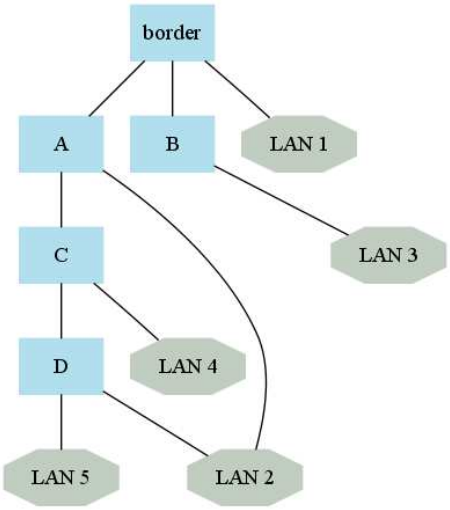

Figure 2.7: Logical Representation of a Network with a Loop

default, the router A will be considered as the main gateway for this network, and D the backup. The routing infrastructure will be configured accordingly, and the routing protocol chosen will take care of sending the traffic to the rig th router.

It may be interesting to be able to define manually a different default gateway than the one determined with the shortest path algorithm. Some load type metrics can also be used to determine that upstream. To do so, we will need to add more information in the network representation. We will discuss this in a later section.

2.4 Multihomed Network

One of the new features brought by IPv6 is network multihoming, where a same subnet can be addressed with two or more different network prefixes. This case represents one of the constraints that the transition algorithm will take. We will take the example of a subnet multihomed with two network prefixes.

As shown in figure 2.8, the two prefixes can be advertised by two routers.

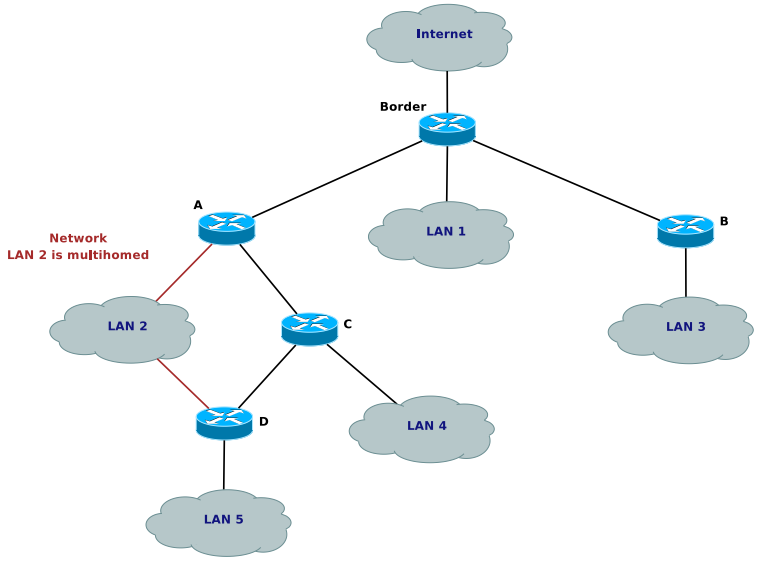

Figure 2.8: Multihomed Network by 2 Routers

In order to generate the logical representation, even if physically the network is the same, it will be represented as two different logical networks, and the loop will thus disappear from the graph, as shown in figure 2.9.

Each network prefix advertised is such seen as the network it addresses with its gateway determined by the shortest path algorithm. However, multihoming can be in that case coupled with redundancy, with D being the backup router for the prefix X and A the one for the prefix Y, as shown in figure 2.10.

It is thus necessary to have the possibility to select manually the upstream for each network, otherwise A will be selected as the default gateway for each subnet. This will permit to let the automatic designation of A as upstream for *LAN 2 Prefix X*, and parameter D as default gateway for *LAN 2 Prefix Y*.

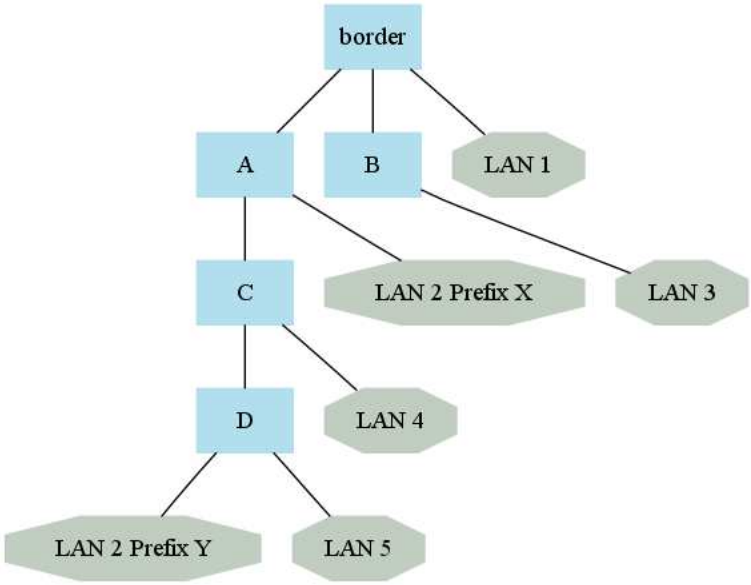

Figure 2.9: Logical Representation of a Multihomed Network by 2 Routers

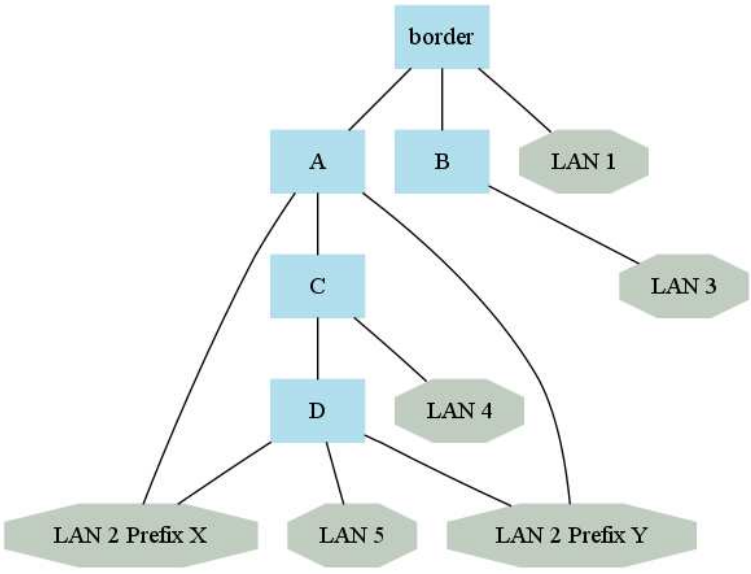

Figure 2.10: Logical Representation of a Multihomed Network by 2 Routers with Redundancy

The prefixes can also be announced by the same router, as we can see in figure 2.11.

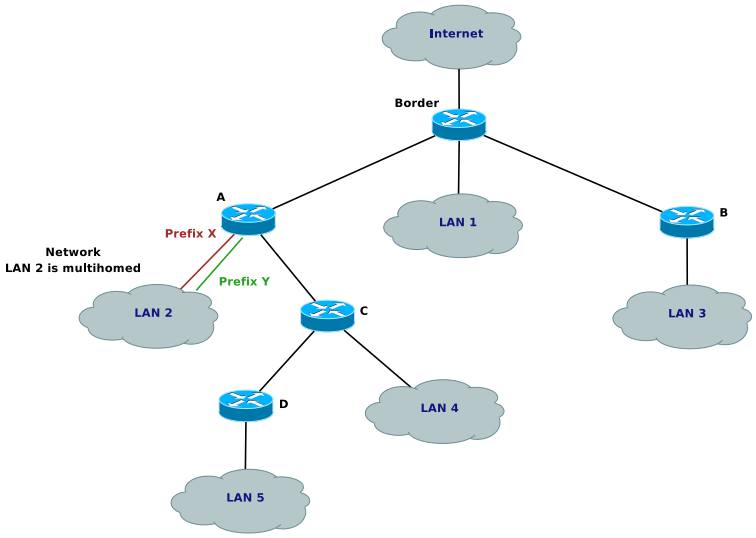

Figure 2.11: Multihomed Network by the same Router

In that case, the network are once again seen as two subnets, but both of them are them are linked only to the router A, which will be their default upstream calculated by the shortest path algorithm, as seen in figure 2.12.

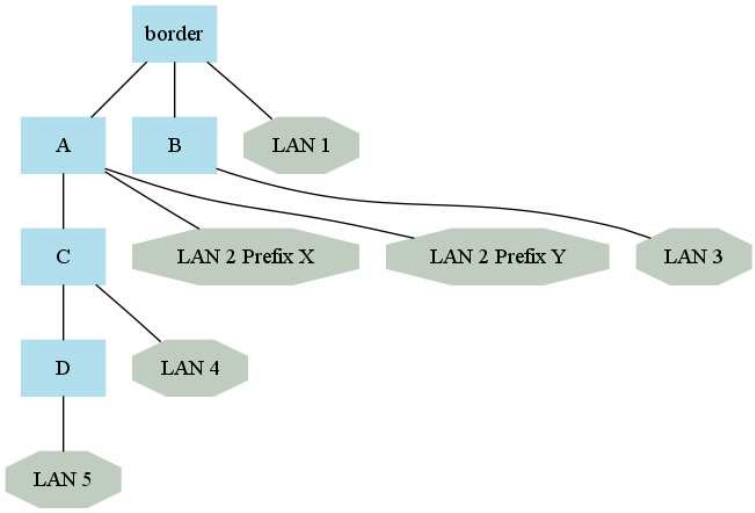

Figure 2.12: Logical Representation of a Multihomed Network by the same Router

If redundancy is added and the networks are also connected to the router D, we have the same representation than in figure 2.10, but no specific constraint or parameter is added, and the gateway for both subnet is determined by the shortest path algorithm, namely A, and D will be the backup.

2.5 Multihoming at Site Level

Multihoming can also be used at the site level. It means that the whole site is multihomed with two different prefixes, usually from 2 ISPs different. In this section, we will not discuss the reasons behind site multihoming, but take a look at the 2 possible situation, where we have 2 ISPs on the same border router, or two border routers, one per ISP.

2.5.1 2 ISPs and 1 border router

In this case, there is only one border router which is connected to two different IPv6 ISPs, each one assigning a different prefix to the site.

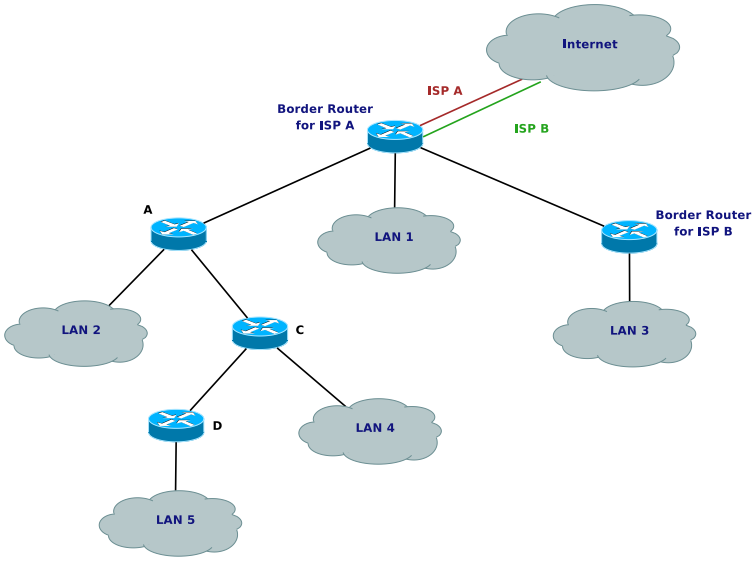

Figure 2.13: Multihomed Site with 1 Border Routers

This does not change anything for the site itself, the logical representation of the topology will be the same that if there was only one ISP and one prefix, as shown in figure 2.2. The difference is in that case that the algorithm will have to be run two times on the same logical representation, one time per ISP. Each iteration of the algorithm will have its own constraints, and the result of the first instance can be used as constraint for the second one.

2.5.2 2 ISPs and 2 border routers

Here, we still have a multihomed site with two different ISPs and prefixes, but each ISP has its own dedicated border router.

Thus, we have two different logical representations of the network, one per ISP, with the corresponding border router as root of the graph. Figure 2.15 shows the logical representation for the ISP A, whereas figure 2.16 shows the one for the ISP B.

The algorithm will be instantiated in each representation with its own constraints. Once again, the result for one ISP can be used as constraint for the other one.

It is also possible that some parts of the site are not multihomed, and belong only to one of the two ISP. In that case, these parts of the topology will not appear in the logical representation of the ISP it does not belong to.

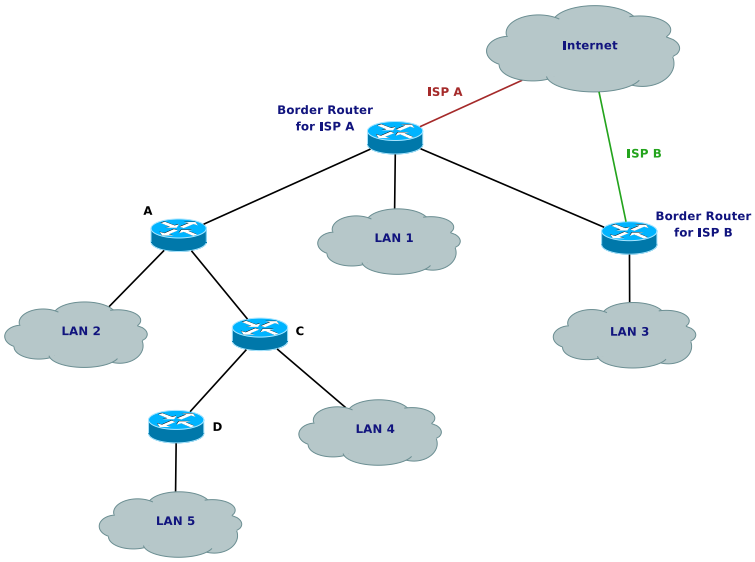

Figure 2.14: Multihomed Site with 2 Border Routers

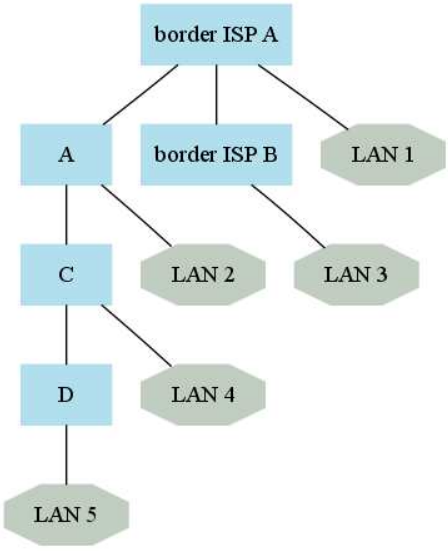

Figure 2.15: Logical Representation for ISP A

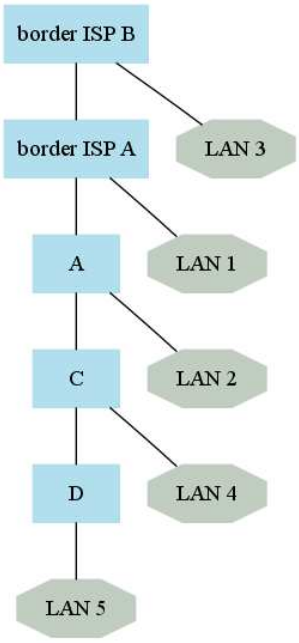

Figure 2.16: Logical Representation for ISP B

2.5.3 Different Border Router

In this case, the border routers managing IPv4 and IPv6 are different, as shown in figure 2.17.

It implies that some network are IPv4 only, whereas others are IPv6 only. The logical representation deduced from this topology will only include the IPv6 parts of the topology, as seen in figure 2.18.

This may raise an issue when automating the configuration and deployment of IPv6. In this situation, the IPv6 border router and the router E do not have an IPv4 address. Automation of the configuration process requires IPv4 addresses on the components to configure in order to connect via Telnet or SSH and push the data. The components that do not have such an address will need manual configuration, or the set up of temporary IPv4 addresses. If temporary addresses are set, it is mandatory to adapt the IPv4 security policy accordingly, to avoid creating security holes. In the other case, the configuration to be pushed to the components will be generated and the administrator will only have to push it into the devices.

2.6 Network with DMZ

In this case, the network is still represented as a hierarchical tree, but the network is also composed of a DMZ, as shown in figure 2.19.

The DMZ is a particular subnet, with specific constraints, in terms of security, addressing... A typical constraint is that the DMZ does not perform address autoconfiguration, but uses static addressing or DHCPv6, or to force a given network prefix on the DMZ or any other subnet. but all these constraints do not impact the logical representation. As shown in figure 2.20, a DMZ is seen as a regular leave in the graph.

This logical representation is not sufficient to specify all the constraint that can be applied to the DMZ. We need more information which need to be specified by another way than the graph. These constraints such a as an autoconf or DHCP flag, the prefix assigned to the DMZ or the addresses assigned to the prefixes, a DMZ flag... need to be specified in addition to the topology, as the subnet requires specific

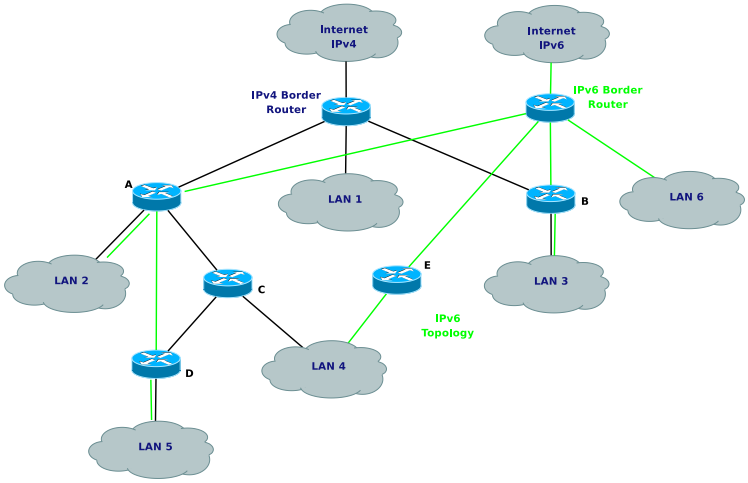

Figure 2.17: Network with Different IPv4 and IPv6 Border Routers

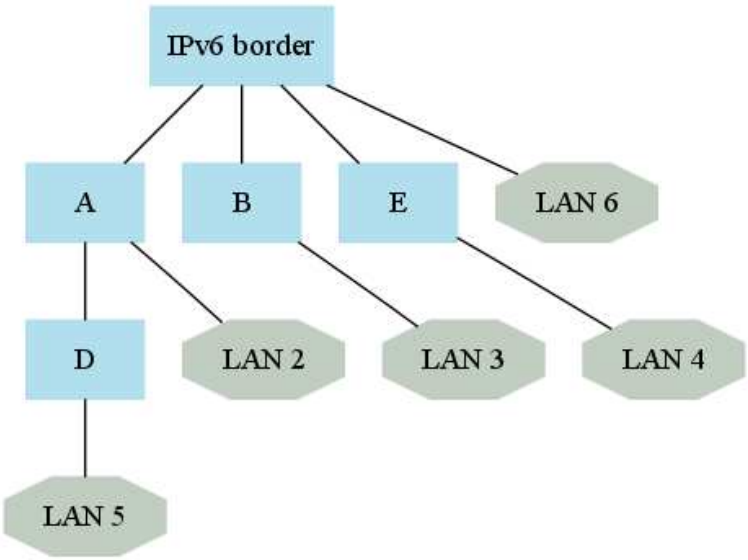

Figure 2.18: Logical Representation of a Network with Different IPv4 and IPv6 Infrastructures

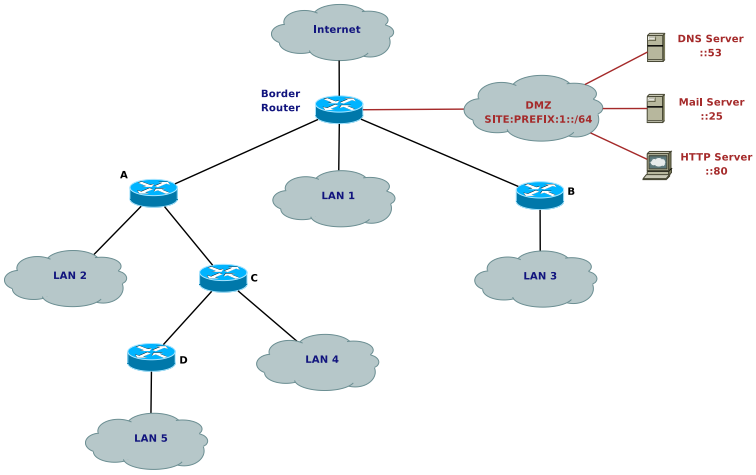

Figure 2.19: Network with a DMZ

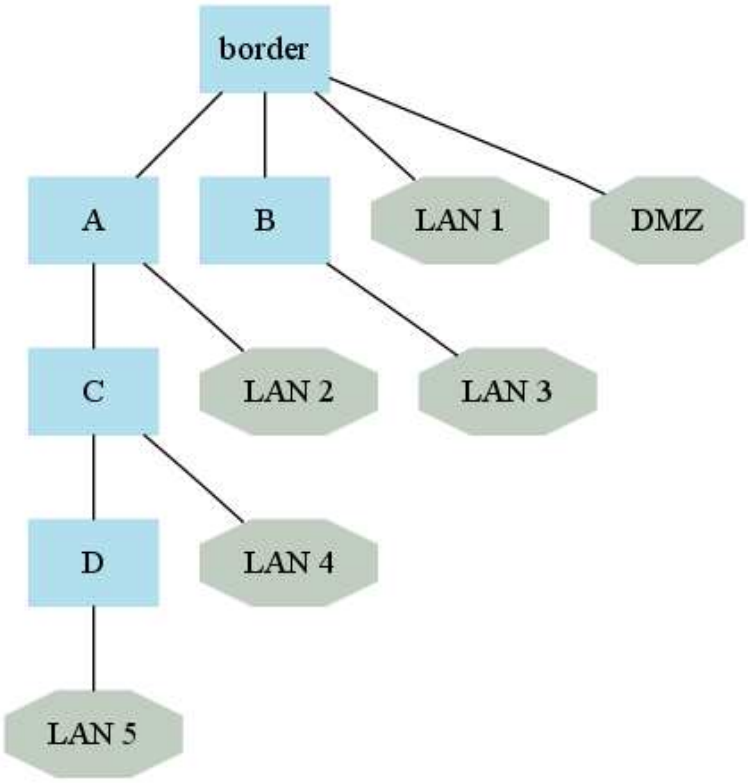

Figure 2.20: Logical Representation of a Network with a DMZ

configuration and a more strong security policy.

2.6.1 VLAN

This time, the specificity is that VLANs have been deployed on the network as seen in figure 2.21.

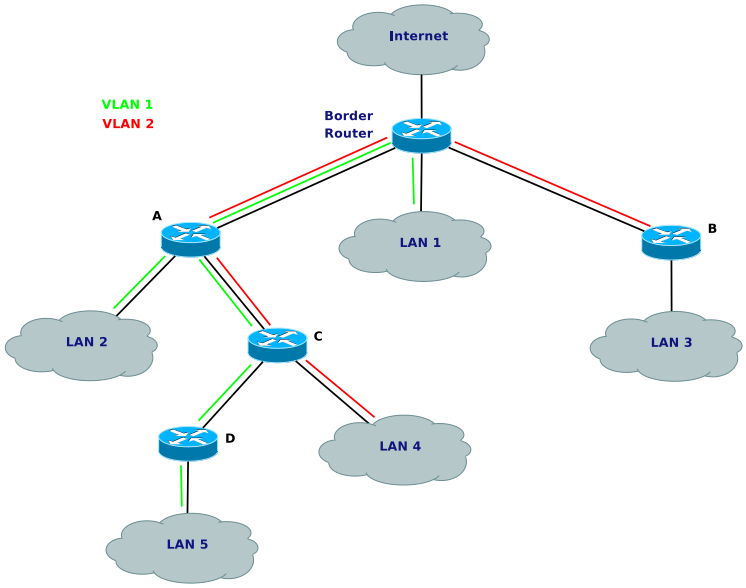

Figure 2.21: Network with VLANs

The VLANs are spread over the whole network, and the addressing plan should respect the definition of these VLANs, while trying to respect as much as possible prefix aggregation, but this may be a difficult constraint.

In the logical representation, each VLAN will be seen as a different link. But for configuration issues, it is important to differentiate these links. The end-points of the links represent the physical or logical physical interface interface. We will use this and put labels on the links to differentiate them as shown in figure 2.22.

The interfaces on the routers follow the Cisco naming convention $< interfacetype >< slot > / < interfaceid > . < vlanid >$. Informations about the VLANs have to be added to this representation, and can be put in the same data input than the information about the DMZ. This may include VLANs definition, information about the interfaces and the VLANs associated, assign a given prefix to a given VLAN...

The informations that are required will be defined when studying this issue later during the project. One of the open questions remaining concerns the network switches, and if they need, for this specific case, to be included in the network representation.

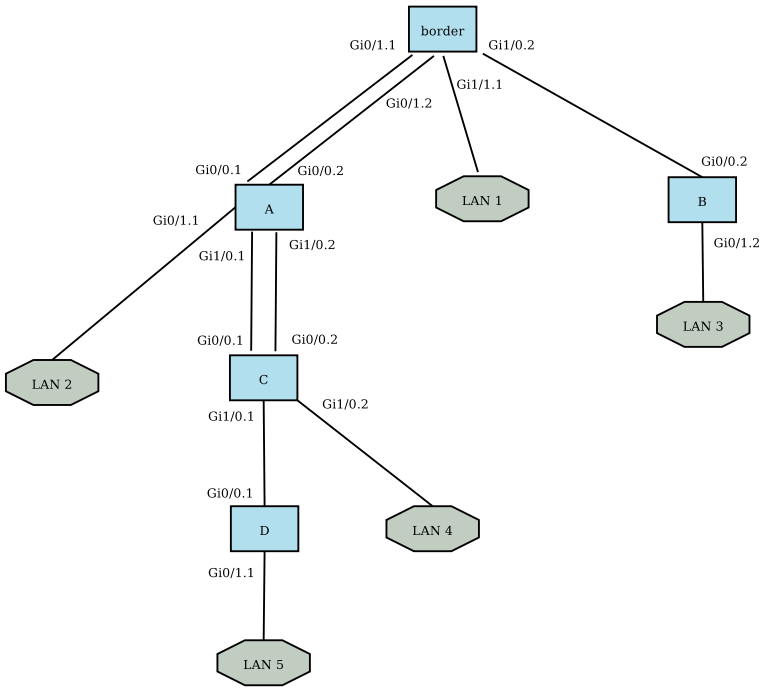

Figure 2.22: Logical Representation of a Network with VLANs

Chapter 3

Pre-requisites to an IPv4 to IPv6 Transition

In this chapter, we will give a non exhaustive set of pre-requisites in order to perform a smooth IPv4 to IPv6 transition. More precisely, we will present the pre-requisites from the point of view of our transition mechanisms. These pre-requisites are the information that are required to perform the transition.

3.1 Connection to the IPv6 World

As we already said, the connection to the IPv6 world will not be discussed in our study. Many other projects already addressed it, and documentation is available very easily.

The connection to the IPv6 world is thus considered as a pre-requisite in this study. We call it pre-requisite even it this operation only takes place in the fourth step of the procedure detailed in the chapter 4 of D1.1.

3.2 Network Topology

This deliverable is presenting the network representation that we will use during our study. In chapter 2, we deduced a logical representation from the actual physical topology.

The knowledge of the physical topology is thus mandatory, and stands for one of the most important pre-requisites for a transition. Administrators usually have a good idea of the topology of their network, but it is mandatory to review it before performing the transition, and audit all components to ensure that they are IPv6 capable.

In chapter 4, we will show how informations can be extracted from the applications usually deployed on networks and that will ease this phase and make it faster.

3.3 Additional Network Related Information

Sole the topology informations are sufficient to deduce the network representation, but, for certain scenarios and in order to perform the configuration, more information are needed. These information can not be set on the graph. During the implementation phase, another input will be needed, basically a configuration file.

3.3.1 Routers

In order to generate the appropriate configuration for the network components, information about the routers interfaces are required. This includes interfaces identifiers to assign IPv6 addresses, and IPv4 addresses to perform remote configuration.

Moreover, to perform the remote configuration, we will also need a valid username and password to connect to the router.

Finally, the router itself must be identified, to differentiate for example a Cisco router and a Quagga based router, in order to generate the appropriate configuration for each device.

3.3.2 VLANs

VLANs are represented as different logical links. But more information are needed, such as the VLAN definition, namely its identifier, and all the constraints that may accompany it, such as a given prefix. Some parameters must be added also in the routers definition. Information about the VLANs and associated interfaces must be added, especially because of the different parameters and possibilities offered by devices from different manufacturers.

Moreover, the switches may need to be included in the representation, and we need to know which port is assigned to which VLAN.

3.4 Constraints

The logical representation does not embed the constraints, but only represents the topology. All the constraints must be identified and defined before generating the configuration, and for some of them before running the addressing. These constraints can not be specified in the graph, but in the same configuration file than the other ones.

In this section we will present a first set of constraints we identified so far. We may add some more as the study advances.

Reserved metric When defining the addressing plan, we sometimes know in advance that we may need in the (near) future more /64 prefixes for network growth (merge with another network, testbed...). Tuning this metric on a router increases the number of /64 prefixes it requests.

Fixed /64 For a given subnet or link in the topology, the administrator can fix a chosen /64 prefix. Usually, this is done for some configuration issues, either locally for service dependencies, or for remote connection with partners, but any other reason can motivate this choice. The algorithms will have to integrate this constraint in the definition of the addressing plane, while making sure to respect prefix aggregation.

Backbone When more than two routers are connected to the same network, it is considered as a backbone (it is just the terminology we chose in our algorithms). The upstream or designated router for that network is determined by using the shortest path to the root algorithm. This feature can also be tuned by using the *force upstream router* constraint.

Multihoming A subnet or a whole site can be multihomed. This constraints permits to specify if a whole site or just a subnet or part of the network is. We need to define how this constraint will be represented in the DOT/XML for the different scenarios defined.

Force upstream router In some topologies, when a loop or in case of multihomed subnets, the administrator may want to use a different root as default upstream than the one chosen with the shortest path algorithm. This constraints makes that possible.

Router-to-router link By default, the transition engine uses /64 networks for links between two routers. A link is thus considered as another subnet. But this solution, if it makes easier the

aggregation, wastes /64 prefixes. RFC 3627 [4] presents a solution where /127 or /126 prefixes can be used for these links. Another option would be to use point-to-point connection by using the Link Local Addresses for example [5]. Before integrating these mechanisms in the algorithms, we will study them, and see how they can be applied applied to avoid wasting /64 prefixes in the context of our approach. The important question we need to answer is: how will these solutions impact prefix aggregation? As aggregation is one strong requirement in our approach, we do not want to compromise it. Another interrogation addresses the links we named backbones, where more than two routers are concerned. Some practical feedback about the solutions used or studied in real cases would be a great help.

Stateless/stateful addressing By default, the transition engine considers that all subnets use Stateless Autoconfiguration. However, sometimes, the administrator may prefer a stateful mechanism with DHCPv6. This constraints permits to tune this feature.

DMZ In networks, especially for enterprises or big organizations, a special subnet called DMZ is used to gather all servers and sensitive services. A DMZ usually implies a stateful addressing mechanism such as DHCPv6 and more complex and stronger security policy. A subnet marked as DMZ in a topology should be addressed consequently by the algorithms.

NAT NAT is not always used only for connection sharing issues, it may be used to hide some information behind this mechanism. The transition to IPv6 must not unhide these information, which means that some more severe security rules need to be applied. This constraint raises one of the big issues that we need to address during this study.

Existing IPv6 address plan For some reasons (redundancy or site multihoming for example), an IPv6 addressing plan for a different prefix may already be present. This addressing plan, and the security policy that is set for this prefix, need to be taken into account when defining the addressing plan and the security policy for the site. We need for this to identify which parts can be reused and in what extend this process can be automated, especially for the security policy, as the needs can be different, depending on the border router used and the other constraints set.

Before integrating all these constraints in the algorithms, the first step will be to study all the new standards released recently, concerning the protocol itself (RFC 4861 [3], RFC 4862 [7], RFC4294 [2] or draft-ietf-v6ops-ipv6-cpe-router-00), some related mechanisms (RFC3627 [4], RFC5072 [5]), and identify the possible changes or new constraints they may raise, or modify the algorithms accordingly. Constraints such as *DMZ* or *NAT* will addressed in parallel with Task 3 as it is part of the security policy study.

Then, as a second step, we will begin running the algorithms on the scenarios, and adding the constraints, in order to update the transition engine.

Chapter 4

Possible Sources of Information

To perform the IPv4 to IPv6 transition, many information about the existing network are required. Most of these information already appear at some place in the management plane. The objective in this chapter is to identify where we can find as much of these data as possible automatically, and generate the software representation of the logical one proposed in this deliverable with as less manual intervention as possible. We identified three main types of information:

- the network topology
- the addressing plane and configuration of the network devices
- the IPv4 security policy

The automatic approach is a strong constraint in our study. In the existing prototype, the network topology and addressing plane, alongside with all required configuration data, are represented in a combination of two file, one in DOT [1] language representing the network topology as a graph, completed by an XML file with all the missing data (addressing plane, configuration of the devices...). Filling these files is not difficult and feasible if the network is not too big. Otherwise, it can be a painful and error prone operation.

We aim at retrieving from the existing IPv4 network and management plane as much information as possible. It seems irrealistic to think that we would be able to get all the information, some data, such as username and passwords to manage remotely the devices for example, is impossible to get, because of security mechanisms such as shadow passwords.

In section 4.1, we will give a non-exhaustive list of tools we studied, and the conclusions we reached, to propose a mechanism discover the required information while minimize the manual intervention of the administrator, and limit it to some details and validation of the output. In section 4.2, we will discuss the interaction between the transition engine and CiscoWorks, while we will explain why the IPv4 security policy is also concerned in section 4.3.

4.1 Automatic Network Discovery

In this section, we present some Open Source or proprietary tools that make possible network topology discovery. This list is not exhaustive, and does not present all the tools we tried. At the end, we give a small conclusion, where we compare the different tools and explain our choice.

[1] http://www.graphviz.org/doc/info/lang.html

4.1.1 Hynetd

Hynetd [2] (Hybrid Network Topology Discovery) is based on a hybrid methodology, that combines active and passive measurements to discover network topologies at router level. The topologies are discovered starting from the IP address spaces to explore and SNMP community names. After specifying the parameters (SNMP community, IP range to scan...), the tools scans the network with ICMP (ping, traceroute), UDP, SNMP, backtrace algorithms, serial link heuristics... The tool also resolves DNS names, ARP tables on the routers via SNMP. As an output, the tools produces several log files (links, routers, subnets, ICMP table, aliases) and one XML file representing the topology discovered. We were very interested in the distinction between routers, subnets and links, as we had followed the same terminology in the logical representation.

The tools aims at guarantying completeness, accuracy and efficiency. This means that the tools aims at discovering the entire topology, without making any mistakes, while minimizing the discovery duration and the traffic overhead.

During our study, we tried two versions. The oldest one was version 0.2.4, which was giving good results in term of accuracy and efficiency, but not in terms of completeness. Several subnets, links or routers where not discovered, because of issues with the SNMP OIDs.

The results are way better with version 0.2.5, which is able to detect the whole topology, with a few mistakes. These mistakes are mainly IPv6 nodes identified as routers whereas they are mere hosts. But by combining the different informations (ICMP and ARP tables, log files...), these mistakes can be solved in most cases.

4.1.2 Cheops-ng

Cheops-ng [3]is a Network management tool for mapping and monitoring a network. It has host/network discovery functionality as well as OS detection of hosts. Cheops-ng has the ability to probe hosts to see what services they are running. On some services, cheops-ng is actually able to see what program is running for a service and the version number of that program. The architecture of the tool relies on a backend agent running on the host, and a GUI that pilots it.

During our tests, cheops-ng did not work very well, crashing often, duplicating some hosts, and its frontend is not very user friendly. Moreover, it did not offer a better view of the network than Hynetd while being slower.

4.1.3 Netdisco

Netdisco [4] is an Open Source web-based network management tool. The target users are large corporate and university networks administrators. Data is collected into a Postgres database using SNMP and presented with a clean web interface using Mason.

Configuration information and connection data for network devices are retrieved via SNMP. Data is stored using a SQL database for scalability and speed. Layer-2 topology protocols such as CDP and LLDP provide automatic discovery of the network topology. Netdisco gets all its data, including topology information, with SNMP polls and DNS queries. It does not use CLI access and has no need for privilege passwords.

Even if the support of CDP and LLDP and other features (locating a host by MAC or IP and show the switch port it lives at) is very interesting, its installation is painful as it has many dependencies, and the facts that it is web based and that it requires a lot of manual interaction does not meet our requirements. It may be a way for the administrator to discover some information that Hynetd did not find automatically and correct the proposed topology and configuration, but it can not be used as the sole discovery tool we will use.

[2] http://www.grid.unina.it/software/TD/
[3] http://cheops-ng.sourceforge.net/
[4] http://netdisco.org/

4.1.4 Zenoss

Zenoss [5] is a monitoring infrastructure that manages the configuration, health and performance of networks, servers and applications through a single, integrated software package. On the opposite to the other tools mentioned so far, even if it also an Open Source application, it remains a commercial one.

Moreover, the user interface is once more a web based, and even if the tool offers several interesting features, it requires more human intervention than what we want. Moreover, the installation process and even the downloading part, for which a registration process is required, are time consuming, while we aim at something fast and easy.

Once again, as it is the case for Netdisco, Zenoss may be a nice source of additional information (even more if it is already deployed in the network), but can not be our first source of information, and may not be worth installing only in the scope of the transition and the collect of the information Hynetd could not retrieve itself.

4.1.5 Homemade Cooking

After trying Hynetd version 0.2.4, we were not satisfied with all the tools we tested, and thought of writing our own Hynetd like scripts by using well known open source software such as ping, traceroute, nmap, snmp, which are often installed on the management station and may be easily interfaced with the transition engine.

We managed to obtain the same kind of information than Hynetd, and encountered the same issues related to the SNMP OIDs. As they are different, depending on the implementation and even sometimes the versions of the frameworks or MIBs, there were many cases to take into account.

We then thought of extending Hynetd, but as we began to look into the code, version 0.2.5 was released, solving most of the problems encountered. Thus, we decided to stick to Hynetd, and use these other tools to correct and complement Hynetd results.

4.1.6 Conclusion

There are a lot of open source software performing automatic network discovery, but most of them require a lot of human intervention, either for the discovery itself, or to correct the results (remove ghosts, duplicates...). As some of them, and especially Hynetd in the context of our study and the requirements we fixed, were giving convincing results, we decided to use Hynetd and other well known tools (nmap, traceroute, ping, snmp...), that are usually installed on the management host, to complement or the information collected. The transition engine will be updated accordingly, and will integrate this automatic discovery.

However, some information will not be discovered automatically, such as the username and password for remote configuration. Moreover, manual intervention will always be required, for validation, or correction/addition of information. To do so, other management tools may be used, such as Nagios or ArpWatch for example. When writing the guidelines and the final version of the report concerning this automatic discovery, we will integrate a list of tools that may help, and the type and location of the concerned data. It may also be of some interest to consider Link Layer Discovery Protocol (LLDP, IEEE standard 802.1AB-2005) and Cisco Discovery Protocol (CDP) protocols, and the NetInventory [6] tool from Bell-Labs.

As this issue is more an engineering problem than a research challenge, we will not focus on it in a first time. We will keep it as a background task, even if it is only finalized after the end of the project.

[5] `http://www.zenoss.com/product/network-monitoring`
[6] `http://www1.bell-labs.com/enablingtech/netinventory.html`

4.2 CiscoWorks

Another option considered is to rely on a management framework such as CiscoWorks. It is a web-based of tools aiming at helping users manage a Cisco-based computer network. It makes possible to get/push configurations on Cisco devices on a network, manage and monitor various aspect of the network, including its topology, fault management... All information related to the addressing plane and topologies are present in this framework. However, it is still unclear how we could interfere with it, as many different options are possible, and some questions need to be answered.

The first option would be to develop the transition engine as a plugin for CiscoWorks. That way, we would benefits from all features of CiscoWorks, including the knowledge of the topology and all the configuration parameters of the devices managed. But this raises an issue concerning networks running non-Cisco devices. Moreover, it is still unclear how and in what extend this plugin integration would be realizable.

The second option would be to use CiscoWorks as another source of information, by taking advantage of the XML serialization offered by the framework to fill missing information, at least for Cisco devices. This relation could be extended, and the transition could also use CiscoWorks to get/push configurations from/to the devices, offering thus more options and a better flexibility (CLI, telnet or SSH in a first time, maybe Netconf or another mechanism in the future without big code updates).

Other questions may be raised during the study, which is why, to answer these interrogations and the ones that will appear, more interaction with Cisco is required on that point.

4.3 IPv4 Security Policy

Finally, the IPv4 Security Policy is another important information to gather before performing the transition. When adding IPv6 to a network or a part of it, it is mandatory to not open new security holes for that new protocol, in order not to compromise the whole infrastructure. Thus, defining a adequate security policy for IPv6 is one of the most important concerns during a transition. Defining this security requires a good knowledge of the protocol and the threats it is facing, and a study of the security policies that are theorically and practically defined and used.

But the most important information to take into account is the IPv4 security policy. The best way to ensure an adequate protection is to reflect the IPv4 policy in the IPv6 one, while adding the answers to the IPv6 only threats. This means that we get the knowledge of the IPv4 security policy, and especially the ACLs placed on all firewall, not only at the networks level, but also locally on host, especially on servers. It is important that a filtered service for IPv4 remains filtered for IPv6. This operation requires an accurate mapping from IPv4 subnets and hosts to IPv6 addresses, internally, but also externally, f a client or provider has access to one of our servers in IPv4 and we want to map this access in IPv6.

This issue will be addressed during Task 3 of this project.

Chapter 5

Conclusion

In this deliverable, we defined a logical network representation that we will use during this study. We also presented 6 topologies and 9 scenarios on which we will focus.

We set the basis of the study by defining the pre-requisites and the assumptions we made for a transition from IPv4 to IPv6, and defined the context in which we placed ourselves. We identified a first set of constraints, standing for some specificities in the existing or the expecting network. Finally, we identified different sources of information that could be taken from the existing IPv4 network.

During the next steps, we will integrate all the aspects and conclusions of this deliverable and of the first task.

Bibliography

[1] S. Deering and R. Hinden. Internet Protocol, Version 6 (IPv6) Specification. RFC 2460 (Draft Standard), December 1998. Updated by RFC 5095.

[2] J. Loughney. IPv6 Node Requirements. RFC 4294 (Informational), April 2006. Updated by RFC 5095.

[3] T. Narten, E. Nordmark, W. Simpson, and H. Soliman. Neighbor Discovery for IP version 6 (IPv6). RFC 4861 (Draft Standard), September 2007.

[4] P. Savola. Use of /127 Prefix Length Between Routers Considered Harmful. RFC 3627 (Informational), September 2003.

[5] S.Varada, D. Haskins, and E. Allen. IP Version 6 over PPP. RFC 5072 (Draft Standard), September 2007.

[6] S. Thomson and T. Narten. IPv6 Stateless Address Autoconfiguration. RFC 2462 (Draft Standard), December 1998. Obsoleted by RFC 4862.

[7] S. Thomson, T. Narten, and T. Jinmei. IPv6 Stateless Address Autoconfiguration. RFC 4862 (Draft Standard), September 2007.